

7-3-00

A

## IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Inventorship..... Douceur et al.  
 Applicant..... Microsoft Corporation  
 Attorney's Docket No. .... MS1-480US  
 Title: Suspension and Reinstatement of Reference Handles

TRANSMITTAL LETTER AND CERTIFICATE OF MAILING

To: Commissioner of Patents and Trademarks,  
 Washington, D.C. 20231

From: Lance R. Sadler (Tel. 509-324-9256; Fax 509-323-8979)  
 Lee & Hayes, PLLC  
 421 W. Riverside Avenue, Suite 500  
 Spokane, WA 99201

The following enumerated items accompany this transmittal letter and are being submitted for the matter identified in the above caption.

1. Specification—title page, plus 34 pages, including 55 claims and Abstract
2. Transmittal letter including Certificate of Express Mailing
3. 7 Sheets Formal Drawings (Figs. 1-10)
4. Return Post Card

Large Entity Status ☒ [x]

Small Entity Status ☐ [ ]

Date: 6/29/00

By: Lance R. Sadler  
 Lance R. Sadler  
 Reg. No. 38,605

CERTIFICATE OF MAILING

I hereby certify that the items listed above as enclosed are being deposited with the U.S. Postal Service as either first class mail, or Express Mail if the blank for Express Mail No. is completed below, in an envelope addressed to The Commissioner of Patents and Trademarks, Washington, D.C. 20231, on the below-indicated date. Any Express Mail No. has also been marked on the listed items.

Express Mail No. (if applicable) EL624352989

Date: 6/29/2000

By: Lori A. Vierra  
 Lori A. Vierra

JC542 U.S. PTO

09/608341

06/29/00

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

**Suspension and Reinstatement of Reference Handles**

Inventor(s):

John Douceur

Yoram Bernet

ATTORNEY'S DOCKET NO. MS1-480US

0629001140 MSI-480US PAT APP DOC

1 **TECHNICAL FIELD**

2 This invention relates to handle administration systems and methods, and  
3 particularly to handle administration systems in which reference handles are used  
4 to access resources that are managed by a resource manager.

5  
6 **BACKGROUND**

7 It is not uncommon for software modules operating on computer systems to  
8 require access to shared resources. For example, a given computer program may  
9 require access to files maintained by a file system, or it may require access to  
10 network connections maintained by a network driver. Network drivers may  
11 require access to information structures maintained by a network packet classifier.  
12 This is a complex arrangement that includes numerous software modules, such as  
13 software drivers requiring access to many shared resources and an access  
14 supervisor that either maintains the resources or at least intercedes when a  
15 software module attempts to access a resource.

16 Intercession by an access supervisor is important for several reasons. One  
17 especially important reason is associated with a situation when a software module  
18 deletes a resource. Specifically, if a first software module deletes a resource, then  
19 other software modules that maintain direct pointers to the resource will be unable  
20 to access or use the resource. This is because their pointers will no longer point to  
21 a valid resource. Attempts have been made to solve this problem by notifying  
22 software modules when a resource deletion occurs. However, this requires  
23 detailed accounting and tracking of software modules and their respective pointers  
24 to the resources. As a result, this process is extremely expensive and very  
25 complex.

00629001140US.PAT APP.DOC

1 Another attempt to solve this problem involves having an access supervisor  
2 intercede when a software module requires access to a particular resource.  
3 Interceding ensures that the particular resource still exists before the software  
4 module is granted access to the particular resource. Typically, this is  
5 accomplished by having the access supervisor, through a handle administrator,  
6 issue a handle to each software module for a particular resource instead of  
7 allowing each software module a direct pointer to that particular resource. The  
8 handle is associated with the resource and is used to refer to the particular resource  
9 when it is desired to be used by the software module. The software module does  
10 not use the handle to directly access the resource. Rather, the software module  
11 presents the handle to the access supervisor, which can then use the handle to  
12 obtain a pointer to the resource for that software module. The process of  
13 converting a handle for a resource into a pointer to that resource is known as  
14 dereferencing.

15 Handle administration systems are typically characterized by having  
16 handles that can assume one of two states—an assigned state and an unassigned  
17 state. When a handle is in the assigned state, the handle administrator has  
18 associated that handle with both a resource and a pointer to the resource. The  
19 handle can then be used by software modules any time they want to obtain a  
20 pointer to the resource. To obtain a pointer to a resource, the software modules  
21 simply present the handle to the access supervisor which then checks to determine  
22 whether the handle is valid. If the handle is valid, then the associated pointer to  
23 the resource can be returned to the software module. If the handle is not valid,  
24 then an appropriate notification to the software module can be generated. When a  
25 handle is in the unassigned state, it is not associated with any resource and thus

cannot be used to access a resource. An assigned handle becomes unassigned when it is "released". A handle can be released when the resource with which it is associated is removed or is no longer available for use by the software modules. Releasing a handle means that the handle can no longer be used to access the resource with which it was formerly associated. Once a handle is released, it is available to be associated with another resource and thereby returned to the assigned state.

It would be very desirable, in some situations, to have the ability to tentatively release a handle. Such a tentatively released handle is still associated with a resource, but it is not considered valid, so it cannot be dereferenced to obtain a pointer to the associated resource. Because the tentatively released handle is not fully released, it is not available to be associated with another resource. This tentatively released handle could then be permanently released into an unassigned state, thus making it available for assignment to another resource, or it could be reinstated into an assigned state that maintains its association with the resource with which it has already been assigned. Presently, however, there are no known handle administration systems that allow this kind of operation.

Accordingly, this invention arose out of concerns associated with improving handle administration systems and methods.

## **SUMMARY**

A handle administration system is described in which software agents receive handles to various resources. The illustrated and described embodiments provide multiple states that can be assumed by the handles. An unassigned state is provided in which handles are not assigned to a particular resource, nor can they

1 be dereferenced into pointers to any resources. In the unassigned state, the  
2 handles are available for assignment to particular resources. An assigned state is  
3 provided in which handles are assigned to a particular resource and can be  
4 dereferenced to obtain a pointer to the resource. A suspended state is provided in  
5 which the handles are assigned to a particular resource but cannot be dereferenced  
6 to obtain a pointer to that resource. Advantageously, a suspended handle can be  
7 reinstated to assume the assigned state.

8 In the described embodiment, there are four actions that can cause  
9 transition between the multiple states. From the unassigned state, a handle can be  
10 assigned so that it assumes the assigned state. When in the assigned state, a  
11 handle can be released so that it assumes the unassigned state. Additionally, when  
12 in the assigned state, a handle can be suspended so that it assumes the suspended  
13 state. Once in the suspended state, a handle can be reinstated so that it assumes  
14 the assigned state. Additionally, in the suspended state, a handle can be released  
15 so that it assumes the unassigned state.

16 Having the suspended state advantageously enables certain operations to be  
17 conducted which, in previous handle systems were impossible to implement. An  
18 exemplary operation is a two-phase commit. In a two phase commit operation,  
19 two or more agents are each asked to suspend a handle associated with a particular  
20 resource (phase 1). If all of the agents can successfully suspend their handles, then  
21 all of the handles are released (phase 2). If any agent is unable to suspend its  
22 handle, all of the handles are reinstated so that they assume the assigned state.

23 In one embodiment, a three state handle system is implemented by  
24 incorporating a suitable field in a handle database that is used to manage the  
25 handles. The incorporated field can be a flag that is set to indicate that a particular

1 handle has been suspended. In another embodiment, no additional database fields  
2 are necessary to implement the three state handle system. Here, a handle database  
3 includes a field for handle values. The handle value is the value that is given to an  
4 agent when it desires access to a resource. In this embodiment, when a handle is  
5 suspended, a value is added to the handle value to yield a resultant value. When  
6 an agent presents the original handle value to access the handle's associated  
7 resource, the handle administrator can check the validity of the presented handle  
8 value by comparing it against the resultant handle value. Since the two compared  
9 handle values do not match, the handle is treated as an invalid handle. To reinstate  
10 a handle from the suspended state into the assigned state, the value that was  
11 originally added to the handle value is subtracted from the handle value to yield  
12 the original handle value. Now, when an agent presents the handle value it will  
13 compare favorably and thus be appropriately treated as a valid handle.

#### 14 15 **BRIEF DESCRIPTION OF THE DRAWINGS**

16 Fig. 1 is an exemplary computer system that is suitable for implementing  
17 the embodiments discussed below.

18 Fig. 2 is a high level block diagram of an exemplary handle administration  
19 system in accordance with the described embodiments.

20 Fig. 3 is a state diagram that describes three exemplary states that handles  
21 can assume in a handle administration system in accordance with the described  
22 embodiment.

23 Fig. 4 is a flow diagram that describes steps in a method in accordance with  
24 the described embodiment.

Fig. 5 is a diagram that illustrates a handle record in accordance with one embodiment.

Fig. 6 is a diagram of a handle database that is used in a handle administration system in accordance with the Fig. 5 handle record.

Fig. 7 is a flow diagram that describes steps in a method in accordance with the described embodiment.

Fig. 8 is a diagram of a handle database in accordance with one embodiment.

Fig. 9 is a flow diagram that describes steps in a suspension method in accordance with the described embodiment.

Fig. 10 is a flow diagram that describes steps in a reinstatement method in accordance with the described embodiment.

## **DETAILED DESCRIPTION**

### **Exemplary Operating Environment**

Fig. 1 shows but one example of a computer 130 that can be used to implement the described embodiments.

Computer 130 includes one or more processors or processing units 132, a system memory 134, and a bus 136 that couples various system components including the system memory 134 to processors 132. The bus 136 represents one or more of any of several types of bus structures, including a memory bus or memory controller, a peripheral bus, an accelerated graphics port, and a processor or local bus using any of a variety of bus architectures. The system memory 134 includes read only memory (ROM) 138 and random access memory (RAM) 140. A basic input/output system (BIOS) 142, containing the basic routines that help to



Computer 130 further includes a hard disk drive 144 for reading from and writing to a hard disk (not shown), a magnetic disk drive 146 for reading from and writing to a removable magnetic disk 148, and an optical disk drive 150 for reading from or writing to a removable optical disk 152 such as a CD ROM or other optical media. The hard disk drive 144, magnetic disk drive 146, and optical disk drive 150 are connected to the bus 136 by an SCSI interface 154 or some other appropriate interface. The drives and their associated computer-readable media provide nonvolatile storage of computer-readable instructions, data structures, program modules and other data for computer 130. Although the exemplary environment described herein employs a hard disk, a removable magnetic disk 148 and a removable optical disk 152, it should be appreciated by those skilled in the art that other types of computer-readable media which can store data that is accessible by a computer, such as magnetic cassettes, flash memory cards, digital video disks, random access memories (RAMs), read only memories (ROMs), and the like, may also be used in the exemplary operating environment.

A number of program modules may be stored on the hard disk 144, magnetic disk 148, optical disk 152, ROM 138, or RAM 140, including an operating system 158, one or more application programs 160, other program modules 162 (such as one or more image synthesizing programs), and program data 164. A user may enter commands and information into computer 130 through input devices such as a keyboard 166 and a pointing device 168. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish,

Computer 130 commonly operates in a networked environment using logical connections to one or more remote computers, such as a remote computer 176. The remote computer 176 may be another personal computer, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to computer 130, although only a memory storage device 178 has been illustrated in Fig. 1. The logical connections depicted in Fig. 1 include a local area network (LAN) 180 and a wide area network (WAN) 182. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets, and the Internet.

When used in a LAN networking environment, computer 130 is connected to the local network 180 through a network interface or adapter 184. When used in a WAN networking environment, computer 130 typically includes a modem 186 or other means for establishing communications over the wide area network 182, such as the Internet. The modem 186, which may be internal or external, is connected to the bus 136 via a serial port interface 156. In a networked environment, program modules depicted relative to the personal computer 130, or portions thereof, may be stored in the remote memory storage device. It will be

1 appreciated that the network connections shown are exemplary and other means of  
2 establishing a communications link between the computers may be used.

3 Generally, the data processors of computer 130 are programmed by means  
4 of instructions stored at different times in the various computer-readable storage  
5 media of the computer. Programs and operating systems are typically distributed,  
6 for example, on floppy disks or CD-ROMs. From there, they are installed or  
7 loaded into the secondary memory of a computer. At execution, they are loaded at  
8 least partially into the computer's primary electronic memory. The invention  
9 described herein includes these and other various types of computer-readable  
10 storage media when such media contain instructions or programs for implementing  
11 the steps described below in conjunction with a microprocessor or other data  
12 processor. The invention also includes the computer itself when programmed  
13 according to the methods and techniques described below.

14 For purposes of illustration, programs and other executable program  
15 components such as the operating system are illustrated herein as discrete blocks,  
16 although it is recognized that such programs and components reside at various  
17 times in different storage components of the computer, and are executed by the  
18 data processor(s) of the computer.

### 19 20 **Exemplary Handle Administration System**

21 Fig. 2 is a high level diagram that shows a handle administration system  
22 that includes a handle administrator 200. The handle administrator 200 can be part  
23 of a resource or access manager that is not specifically shown. The handle  
24 administrator can be implemented in any suitable hardware, software, firmware or  
25 combination thereof. A plurality of different agents 202, 204, and 206 are

1 provided and are consumers of resources 208, 210, and 212. The agents are  
2 typically software modules, such as dynamic link libraries (DLLs), that require  
3 access to the resources. Resources 208, 210, and 212 can be any resources for  
4 which handles are typically used. Exemplary resources include files, data  
5 structures, or objects that are manipulated by the software modules. The agents  
6 202-206 may require access (either sole or shared) to one or all of the resources.

7 Handle administrator 200 generates and validates handles to provide to the  
8 agents when the agents desire access to a resource. The handle administrator 200  
9 uses the handles to efficiently manage access to the resources 208, 210, and 212.  
10 Typically, the handle administrator uses a handle database to manage various  
11 handles that can be used to access resources. To issue a handle, the handle  
12 administrator typically receives a call from a resource manager that includes a  
13 pointer to a resource. The handle administrator places the resource pointer in the  
14 handle database and associates a handle with the resource pointer. The handle  
15 administrator then returns the handle to the resource manager. Thereafter, the  
16 handle is used to access the resource pointer which, if valid, can be used to access  
17 the resource. When a particular resource is accessed by an agent, the agent  
18 presents the handle to the resource manager or handle administrator. The handle  
19 administrator ensures that the handle is still valid and if so, dereferences the  
20 handle to obtain the associated resource pointer for the agent to use in accessing  
21 the resource. If the handle is invalid, e.g. the associated resource has been  
22 removed, then the handle administrator can take the appropriate steps to ensure  
23 that the agent is notified. This might involve returning a null pointer to the agent.

24 Advantageously, the described handle administrator can suspend handles.  
25 When a handle is suspended, its state is changed from that of "assigned" to that of

1 “suspended”. A suspended handle can be considered as a tentatively released  
2 handle. A tentatively released handle is capable of being reinstated to an assigned  
3 state so that it can be validly dereferenced into a pointer to the same resource.  
4 When a handle is in the suspended state, it may still be associated with its resource  
5 but is incapable of being dereferenced into a pointer to that resource. The result is  
6 that any agent that requests access to a resource using a suspended handle is not  
7 given access to that resource. Once, however, a suspended handle is reinstated,  
8 agents can access the associated resource using the same handle as before.

### 9 10 **Suspended State Handle System**

11 Fig. 3 is a state diagram that describes each of three states that a handle can  
12 have in accordance with the described embodiment. The handle states are  
13 managed by the handle administrator 200 (Fig. 2). Transitions between the states  
14 are accomplished by software routines that cause the handle administrator to  
15 manipulate a handle database that maintains a table of the handles. The illustrated  
16 states include an unassigned state 300, an assigned state 302, and a suspended  
17 state 304. The handle administrator is configured to assign, release, dereference,  
18 suspend, and reinstate handles that are associated with various resources.

19 When a handle is in the unassigned state 300, it can be placed into the  
20 assigned state 302 by an assign routine that assigns it to a particular resource and  
21 associates a pointer to that resource with the handle. The handle can then be  
22 provided to any agents that desire to access the associated resource. When a  
23 handle is in the assigned state 302, it can be placed into the unassigned state 300  
24 by a release routine that marks the handle as invalid so that it can no longer be  
25 dereferenced to obtain a resource pointer. Advantageously, when a handle is in the

1 assigned state 302 it can be placed into the suspended state 304 by a suspend  
2 routine. In the suspended state 304, a handle can be thought of as being  
3 tentatively released. That is, when a handle is suspended, it can still be bound to a  
4 particular resource, but it cannot be validly dereferenced into a pointer to that  
5 resource. Thus, any agents that present a handle to the handle administrator when  
6 the handle is in the suspended state will not receive a pointer to the associated  
7 resource. A handle can be returned to the assigned state 302 from the suspended  
8 state 304 by a reinstate routine. A handle that is in the suspended state 304 can be  
9 placed into the unassigned state 300 through a release routine.

10 Advantages of having a suspended state for handles include the support of a  
11 two-phase commit operation. A two-phase commit operation is useful in the  
12 following context. Assume that there are a number of agents that hold handles to  
13 various resources, such that different agents may hold different handles to the  
14 same resource. Assume also that a managerial component desires to eliminate a  
15 handle to a particular resource because that resource might not be further needed.  
16 An agent can be requested to release one of its handles, but if the agent is still  
17 using the indicated handle or resource, the agent will reject the request and not  
18 release the handle. Having the suspended state 304 enables a plurality of different  
19 agents to be requested to release their handles in a single atomic action. This  
20 means that either all of the agents will release their handles if the handles are not  
21 being used by any of the agents. Alternately, none of the agents will release their  
22 handles if any one of the agents is still using its handle. This result is achieved by  
23 a two-phase commit operation.

## **Two-Phase Commit Operation**

Fig. 4 shows a flow diagram that describes steps in an exemplary two-phase commit operation. The goal of this operation is to have all agents release a set of handles in a single atomic action. Step 400 requests each agent to suspend a particular handle. This step can be implemented by a managerial component calling multiple different agents. Each agent responds to the request by examining its handles to see if the particular handle is in use. If the particular handle is in use, the agent refuses to suspend the particular handle and notifies the managerial component of its refusal. If the particular handle is not in use, the agent calls the handle administrator 200 or resource manager to suspend the particular handle, and the agent notifies the managerial component that it complied with the suspension request. Step 402 determines whether the particular handles have been suspended by each and every agent. If all of the agents have successfully suspended their handles, step 404 orders each agent to release its particular handle. The managerial component can be certain that all agents are able to comply with this order, because all agents have successfully suspended their particular handles, so those particular handles are guaranteed not to be in use. Each agent responds to the order by calling the handle administrator 200 or resource manager to release its particular handle, at which time it becomes unassigned. When a handle is unassigned, it is available for assignment to other resources. If, on the other hand, any of the agents reports back that it cannot or will not suspend its particular handle, then step 406 reinstates all of the particular handles to the assigned state.

## **Exemplary Implementations**

There are likely many ways to implement the above-described suspended state in various handle administration systems. The discussion below describes but two ways that this can be done. The two examples given below are given for exemplary purposes only and are not intended to limit the scope of the claimed subject matter. The first-described implementation makes use of a modified handle database structure, while the second-described implementation can make use of an existing handle database structure.

### **Use of Suspended Flag**

Fig. 5 shows a table 500 that describes an exemplary data structure called a handle record. An array of these data structures is resident on a computer-readable medium and is used by the handle administrator to manage handles. Each of the handle records describes one particular handle in the handle record array.

Each handle record includes three fields 502 (handle), 504 (source) and 506 (suspended). Field 502 is a data item of type Handle that holds the value of the handle described by the record. Field 504 is a data item of type Resource\* (pointer to Resource) that points to the resource associated with the handle. Field 506 is a data item of type bool (Boolean flag) that indicates whether the handle has been placed into a suspended state. This flag can then be checked, for example, by the dereference routine to ensure that the handle has not been suspended before dereferencing the handle into a pointer for a resource.

Fig. 6 shows an exemplary elementary handle database 600 that includes an array of four handle records. Each record is identified by a serial index 602, and each record contains the three fields described in table 500: handle 604, resource



1 606, and suspended 608. The serial index 602 indicates the index value or  
2 database location for a particular handle record. The handle field 604 holds a  
3 handle value for a particular handle record. The resource field 606 holds a pointer  
4 value for a particular resource that is associated with the handle value, and the  
5 suspended field 608 holds a value that indicates whether a particular associated  
6 handle has been suspended. In some systems, when the handle database is  
7 initialized, the handle values are set equal to the index of that record and the  
8 resource pointer is set to a null value. Additionally, the value in the suspended  
9 column can be set to an initial value. In the present example, there are four handle  
10 records in the handle database. The records are numbered 0-3. The handle values  
11 are set equal to the index values. In this particular example, the handle  
12 corresponding to the first index location is assigned to resource "A", the handle  
13 corresponding to the second index location is assigned to resource "B", the handle  
14 corresponding to the third index location is assigned to resource "C", and the  
15 handle corresponding to the fourth index location is assigned to resource "D". The  
16 handle in the fourth record has been suspended.

17 When an agent presents a handle, i.e. handle value, to the handle  
18 administrator 200 (Fig. 2) to access a resource, the handle administrator takes the  
19 handle and determines whether the handle is valid. In this example, the handle  
20 administrator first ascertains the location of the pertinent handle record in the  
21 handle database. In this case, assume that an agent presents a handle value of 0 to  
22 the handle administrator. The handle administrator then locates the pertinent  
23 handle record -- here the handle record corresponding to an index of 0 -- and  
24 checks to determine whether the handle value that was presented by the agent is  
25 valid. The handle administrator also checks to ascertain whether the handle is

1 suspended. To ascertain the validity of a handle or handle value, the handle  
2 administrator might simply compare the handle value presented by the agent with  
3 the handle value that is present in the handle field 604 for the handle of interest. If  
4 the values match, then the handle is valid. To ascertain whether the handle is  
5 suspended, the handle administrator can simply check the status of the suspended  
6 field 608 for the handle record of interest. If the handle is valid and not  
7 suspended, then the handle can be dereferenced into a pointer to the associated  
8 resource. Otherwise, the handle is not dereferenced into a pointer.

9 Fig. 7 shows a flow diagram that describes steps in a dereferencing routine  
10 in accordance with this embodiment. Step 700 receives a handle value from one  
11 or more agents. Step 702 determines whether the handle value is valid. For  
12 example, if a handle has been placed into an unassigned state by being released,  
13 then it is not a valid handle that can be used to access a resource. One way of  
14 releasing a handle is to enter a value in the handle value field that is different from  
15 the present handle value. When the handle administrator compares handle values,  
16 the comparison is not favorable and thus the handle administrator will know that  
17 the handle presented by the agent is invalid. If the handle values do not match,  
18 thereby indicating an invalid handle, step 704 can return a null pointer to the  
19 agent. It will be appreciated that any suitable action can be taken by the handle  
20 administrator when a handle is determined to be invalid. Returning a null pointer  
21 constitutes but one suitable action.

22 Assuming that the handle value is determined to be valid, step 706  
23 determines whether the handle has been suspended. In this particular example,  
24 this step is implemented by simply checking the value of the suspended flag in the  
25 handle record that is associated with handle of interest. If the flag has a

1 predetermined value that indicates a suspended handle, then step 706 can branch  
2 to step 704 and return a null pointer. In this example, a suspended handle is still  
3 associated with a particular resource—it is just treated as an unassigned handle for  
4 purposes of dereferencing. If, on the other hand, step 706 determines that the  
5 handle is not suspended, step 708 returns the appropriate resource pointer to the  
6 agent.

7 Although this approach can work well in many handle administration  
8 systems, there may be times when the additional processing overhead of checking  
9 the suspended flag, and the additional memory consumed by the suspended flag in  
10 each record can be avoided.

### 11 12 **Use of Special Value for Handle Value**

13 In one embodiment, a handle can be indicated as suspended by  
14 manipulating the handle value with a special value. For example, Fig. 8 shows an  
15 exemplary handle database or table that is similar to the handle database of Fig. 6,  
16 except that it does not contain a suspended flag field. Here, each of the handles  
17 that correspond to index entries 0-3 are assigned to respective resources A-D.  
18 Each of the handle values for index entries 0-2 have been initialized to the value of  
19 the index. Notice that the handle value for index value 3 is  $(3 + 2^{10})$ . This handle  
20 value was formerly equal to 3 (i.e. the index value) before the handle was  
21 suspended. This special value now indicates that the handle that corresponds to  
22 index value 3 is suspended. In this particular example, a handle is suspended by  
23 adding  $2^{10}$  to the original handle value. When a handle is suspended, in this  
24 example, it is still associated with a particular resource. It cannot, however, be  
25 dereferenced into a resource pointer for that resource.

1 Consider, for example, what happens when an agent presents a handle value  
2 of 3 to the handle administrator. The handle administrator locates the appropriate  
3 index value that corresponds to the handle value and then determines whether the  
4 handle value provided by the agent matches the handle value in the database.  
5 Here, since 3 does not match  $(3 + 2^{10})$ , the handle administrator treats the handle  
6 as if it is unassigned. This means that the agent that presented the handle value  
7 does not get access to the resource that is associated with that handle value. To  
8 reinstate a handle, the special value that was added to the handle value is simply  
9 subtracted from the value that occupies the handle value field for the handle of  
10 interest. In the described embodiment, the particular handle value manipulations  
11 that enable a handle to be suspended and reinstated are addition and subtraction  
12 respectively. It is to be understood that other handle value manipulations could  
13 take place without departing from the spirit and scope of the claimed subject  
14 matter.

15 In particular implementations, characteristics of the special value that is  
16 used to indicate that a handle is suspended include that the value should be a value  
17 that is larger than the table size of the index table or handle database (in this  
18 example the table size is 4). This value should also take into account the fact that  
19 some handle databases can grow and shrink in size over the course of time. An  
20 exemplary handle database that does just that is described in U.S. Application  
21 Serial No. 09/103334, entitled "Generation and Validation of Reference Handles,  
22 filed on June 23, 1998, which is assigned to the assignee of this application, the  
23 disclosure of which is incorporated by reference herein. In addition, the special  
24 value should be a fixed value so that the handle can be reinstated through a simple  
25 procedure such as a subtraction operation described above. In this particular

1 example, a handle can be reinstated by simply subtracting  $2^{10}$  from the value now  
2 held in the handle value field to yield the original handle value of 3. In addition,  
3 in some handle administration systems it is particular advantageous to use a power  
4 of 2 as the special value for internal purposes.

5 Fig. 9 shows a flow diagram that describes steps in a handle suspension  
6 method in accordance with the described embodiment. Step 900 determines that a  
7 particular handle is to be suspended. This might be done when a resource with  
8 which the handle is associated is desired to be eliminated as mentioned above.  
9 Step 902 suspends the handle by adding a value to the handle value that is  
10 contained in the handle database for the handle of interest.

11 Fig. 10 shows a flow diagram that describes steps in a handle reinstatement  
12 method in accordance with the described embodiment. Step 1000 determines that  
13 a particular handle is to be reinstated. One reason for reinstating a handle might  
14 be the failure of the first phase of a two-phase commit operation. Step 1002  
15 reinstates the handle by subtracting the value (i.e.  $2^{10}$ ) that was originally added to  
16 the handle value to suspend it. This returns the handle value to its original value  
17 and places it in an assigned state.

18 The above described process greatly facilitates the manner in which handles  
19 can be suspended and reinstated. This approach is particularly useful in the  
20 context of a specific handle administration system that is described in U.S. Patent  
21 Application Serial No. 09/103334. For the sake of brevity, the specific operation  
22 of that system is not described herein. In the context of the handle administration  
23 system disclosed in the referenced application, the special value that is utilized to  
24 suspend handle is known as the handle range step. The handle range step is a  
25 special large value that is a power of 2. This value is typically used in the

described system to automatically revoke handles in a system-wide manner. The described system periodically revokes all handles whose values are greater than or equal to a value known as the handle base and less than the handle base plus the handle range step. By using the handle range step as the special value added to a handle value to indicate suspension, several important properties of the described system are preserved: 1) The assign, release, expand, and contract procedures will treat the record as though it is assigned, so the record will be preserved for possible future reinstatement. 2) The dereference procedure will treat the record as though it is invalid, so it will refuse to dereference the handle to a pointer to the associated resource. 3) The expand and contract routines will properly locate the handle record when the database is expanded and contracted. 4) The handle revocation routine will revoke suspended handles at approximately the same time as it would revoke the handle if it were not suspended. The use of a handle range step will be understood by those familiar with the described handle administration system and is not discussed in detail here. In connection with using the handle range step to suspend handles in the referenced disclosure, a few modifications of that system should be implemented. First, in the referenced handle administration system, the handle base is initialized to 0. In the present case, the handle base should be initialized to the record array size minus the handle range step. Additionally, in the referenced handle administration system there is a process for revoking so-called ancient handles. There, a revocation range is described as a range from the handle base to the handle base plus the handle range step. In the present context, the revocation range is redefined as the range from the handle base to the handle base plus two times the handle range step.

1 Additionally, in the referenced handle administration system, the release  
2 routine checks to determine whether the handle value that is passed in by the agent  
3 matches the handle value in the handle record. If it does not, then the release  
4 operation is aborted since the handle passed to the release routine is not currently  
5 assigned. In the present context, the release routine checks whether the handle  
6 value that is passed to the release routine matches the handle value in the handle  
7 record (indicating an assigned handle) or whether it matches the handle value in  
8 the handle record minus the handle range step (indicating a suspended handle). If  
9 either of the values match the value passed to the release routine, then the handle  
10 can be released.

### 11 Conclusion

12  
13 The described embodiments advantageously provide for a three-state  
14 handle administration system. The suspended state enables handles to be  
15 tentatively released from the assigned state. Suspended handles can be reinstated  
16 to their previously assigned state or permanently released into the unassigned  
17 state. Having this flexibility greatly facilitates the use of handle administration  
18 systems. For example, using the above described suspended state, two-phase  
19 commit operations can now be implemented whereas before, with other handle  
20 administration systems, they could not. Other advantages will be apparent to those  
21 of skill in the art.

22 Although the invention has been described in language specific to structural  
23 features and/or methodological steps, it is to be understood that the invention  
24 defined in the appended claims is not necessarily limited to the specific features or  
25

1 steps described. Rather, the specific features and steps are disclosed as preferred  
2 forms of implementing the claimed invention.  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25



1 **CLAIMS**

2       **1.** A multi-state handle administration system in which handles are  
3 capable of assuming states comprising:

4           an unassigned state in which a handle is not assigned to a particular  
5 resource;

6           an assigned state in which a handle is assigned to a particular resource and  
7 can be dereferenced to obtain a pointer to the resource; and

8           a suspended state in which a handle is assigned to a particular resource but  
9 cannot be dereferenced to obtain a pointer to that resource.

10  
11       **2.** The handle administration system of claim 1, wherein handles can be  
12 suspended from the assigned state into the suspended state.

13  
14       **3.** The handle administration system of claim 1, wherein handles can be  
15 reinstated from the suspended state into the assigned state.

16  
17       **4.** The handle administration system of claim 1, wherein handles can be  
18 released from the suspended state to the unassigned state.

19  
20       **5.** The handle administration system of claim 1, wherein handles can be  
21 released from both the suspended and the assigned states to the unassigned state.

1           6.    The handle administration system of claim 1 further comprising a  
2 handle database that is configured to contain indicia that indicate whether a handle  
3 is assigned, unassigned or suspended.  
4

5           7.    The handle administration system of claim 6, wherein said indicia  
6 comprises a field for holding a value that indicates that a handle is in a suspended  
7 state.  
8

9           8.    The handle administration system of claim 6, wherein said indicia  
10 comprises a handle value field for holding a value that indicates that a handle is in  
11 a suspended state.  
12

13           9.    A handle administrator configured to manage handles that are  
14 associated with resources, the handle administrator being configured to place the  
15 handles in one of more than two possible states which affect whether a handle can  
16 be dereferenced to provide a pointer to the resource with which the handle is  
17 associated.  
18

19           10.   The handle administrator of claim 9, wherein one of the states  
20 comprises a suspended state in which the handle is associated with a particular  
21 resource but cannot be dereferenced into a pointer to that resource.  
22  
23  
24  
25

1           **11.**   The handle administrator of claim 10, wherein two of the states  
2 comprise:

3           an assigned state in which a handle is associated with a resource and can be  
4 dereferenced to provide a pointer to that resource; and

5           an unassigned state in which the handle is not associated with any resources  
6 and cannot be dereferenced to provide a pointer to any of the resources.  
7

8           **12.**   The handle administrator of claim 11, wherein the handle  
9 administrator is configured to suspend handles from an assigned state into a  
10 suspended state.  
11

12           **13.**   The handle administrator of claim 11, wherein the handle  
13 administrator is configured to reinstate handles from a suspended state into an  
14 assigned state.  
15

16           **14.**   The handle administrator of claim 11, wherein the handle  
17 administrator is configured to release handles from the suspended state to the  
18 unassigned state.  
19

20           **15.**   The handle administrator of claim 11, wherein the handle  
21 administrator is configured to release handles from the suspended state and the  
22 assigned state to the unassigned state.  
23

24           **16.**   A handle administration system comprising:  
25 one or more computer-readable media; and

software code embodied on the computer-readable media which is configured implement a handle administration system that comprises:

an unassigned state in which a handle is not assigned to a particular resource;

an assigned state in which a handle is assigned to a particular resource and can be dereferenced to obtain a pointer to the resource; and

a suspended state in which a handle is assigned to a particular resource but cannot be dereferenced to obtain a pointer to that resource.

17. A resource management system configured to manage resources comprising:

one or more resources that can be consumed by one or more agents; and

a handle administrator configured to administer a handle system in which handles to the one or more resources are provided to the agents and can be dereferenced into pointers to the one or more resources, the handle system comprising more than two states for a handle, the states comprising:

an assigned state in which a handle is associated with a resource and can be dereferenced into a pointer to that resource;

an unassigned state in which the handle is not associated with a resource and cannot be dereferenced into a pointer to any resources; and

a suspended state in which the handle is associated with a resource but cannot be dereferenced into a pointer to any resources.

1           **18.**    The resource management system of claim 17 further comprising  
2 one or more agents that are consumers of one or more resources.

3  
4           **19.**    The resource management system of claim 17, wherein the handles  
5 can be suspended from the assigned state to the suspended state.

6  
7           **20.**    The resource management system of claim 17, wherein the handles  
8 can be reinstated from the suspended state to the assigned state.

9  
10          **21.**    The resource management system of claim 17, wherein the handles  
11 can be released from the suspended state to the unassigned state.

12  
13          **22.**    The resource management system of claim 17, wherein the handles  
14 can be released from the suspended and the assigned states to the unassigned state.

15  
16          **23.**    A resource management system comprising:  
17 at least one computer readable media; and  
18 one or more handle records resident on the media, each record having  
19 fields, one of the fields being configured to provide indicia of whether a handle is  
20 in a state other than an assigned state and an unassigned state.

21  
22          **24.**    The resource management system of claim 23, wherein the other  
23 state comprises a suspended state in which a handle can be associated with a  
24 particular resource, but cannot be dereferenced to obtain a pointer to that resource.





1           **37.**    The method of claim 36, wherein said indicating comprises setting a  
2 flag to indicate that the handle is suspended.

3  
4           **38.**    The method of claim 36, wherein said indicating comprises  
5 manipulating a handle value in the handle database to indicate that the handle is  
6 suspended.

7  
8           **39.**    One or more computer-readable media having computer-readable  
9 instructions thereon which, when executed by one or more computers, cause the  
10 computers to implement the method of claim 36.

11  
12           **40.**    A method of managing resources comprising:  
13 receiving a handle value;  
14 determining whether the handle value is suspended; and  
15 if the handle value is not suspended, returning a reference pointer that  
16 points to a resource with which the handle is associated.

17  
18           **41.**    The method of claim 40 further comprising determining whether the  
19 handle is valid and returning a reference pointer only if the handle is valid and not  
20 suspended.

21  
22           **42.**    The method of claim 41 further comprising returning a null pointer  
23 if the handle is invalid or suspended.



1           **43.**    The method of claim 40 further comprising returning a null pointer  
2 if the handle is suspended.

3  
4           **44.**    The method of claim 40 wherein the handle has more than two states  
5 into which it can be placed, the states comprising:

6                an assigned state in which the handle is associated with a resource and can  
7 be dereferenced to obtain a pointer to that resource;

8                an unassigned state in which the handle is available for assignment to a  
9 particular resource; and

10               a suspended state in which the handle is associated with a particular  
11 resource but cannot be dereferenced to obtain a pointer to that resource.

12  
13           **45.**    The method of claim 44 further comprising reinstating a handle  
14 from the suspended state to the assigned state.

15  
16           **46.**    The method of claim 44 further comprising releasing a handle from  
17 the suspended state to the unassigned state.

18  
19           **47.**    One or more computer-readable media having computer readable  
20 instructions thereon which, when executed by one or more computers, cause the  
21 computers to:

22                receive a handle value;

23                determine whether the handle value is suspended; and

24                return a reference pointer that points to a resource with which the handle is  
25 associated, if the handle is not suspended.

1  
2       **48.**    The computer-readable media of claim 47, wherein the instructions  
3 cause the computers to suspend an assigned handle, the suspended handle being  
4 associated with a particular resource but being unable to be dereferenced to obtain  
5 a pointer to that resource.

6  
7       **49.**    A method of managing resources comprising:  
8        defining a handle database that contains information pertaining to a  
9 plurality of handles, the information comprising a handle value; and  
10       suspending a handle by adding a value to the handle value.

11  
12       **50.**    The method of claim 49 further comprising reinstating the  
13 suspended handle by subtracting the value from the handle value.

14  
15       **51.**    The method of claim 49 further comprising releasing the handle.

16  
17       **52.**    The method of claim 49, wherein said value comprise a handle  
18 range step value.

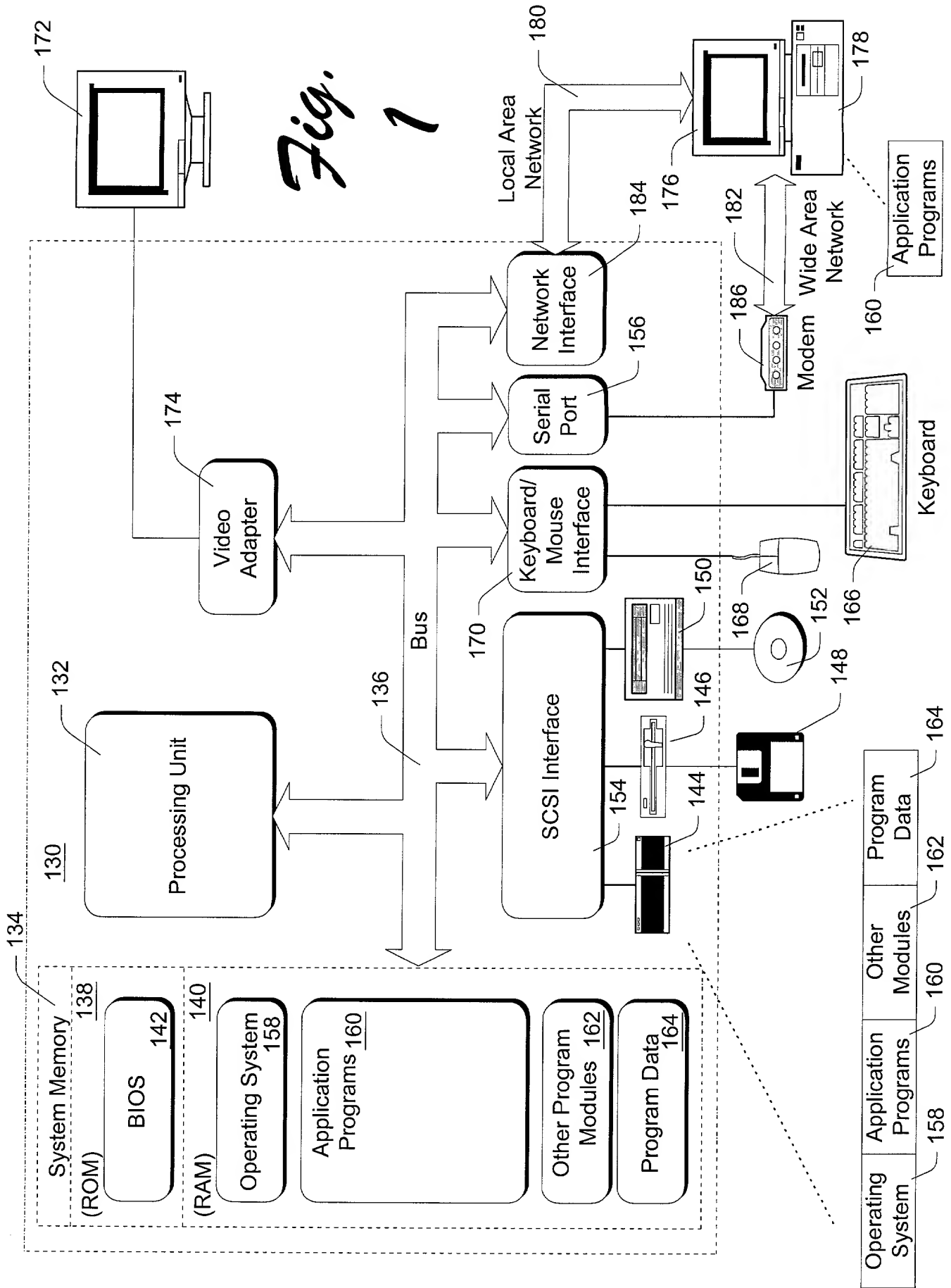
19  
20       **53.**    The method of claim 49, wherein said value comprises a power of 2.  
21  
22  
23  
24  
25

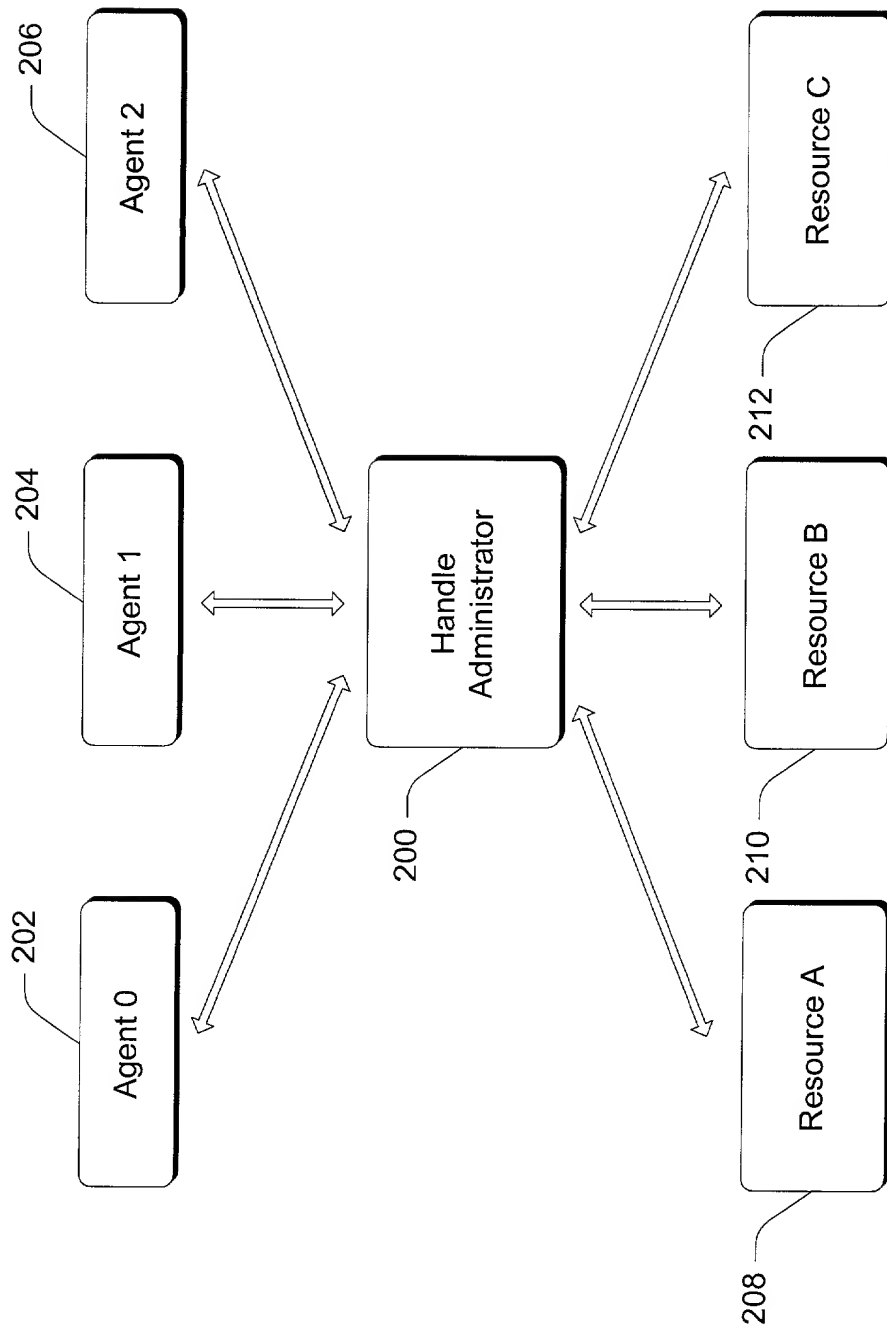


[illegible]

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

Fig. 1



*Fig. 2*



005290" T4E80950

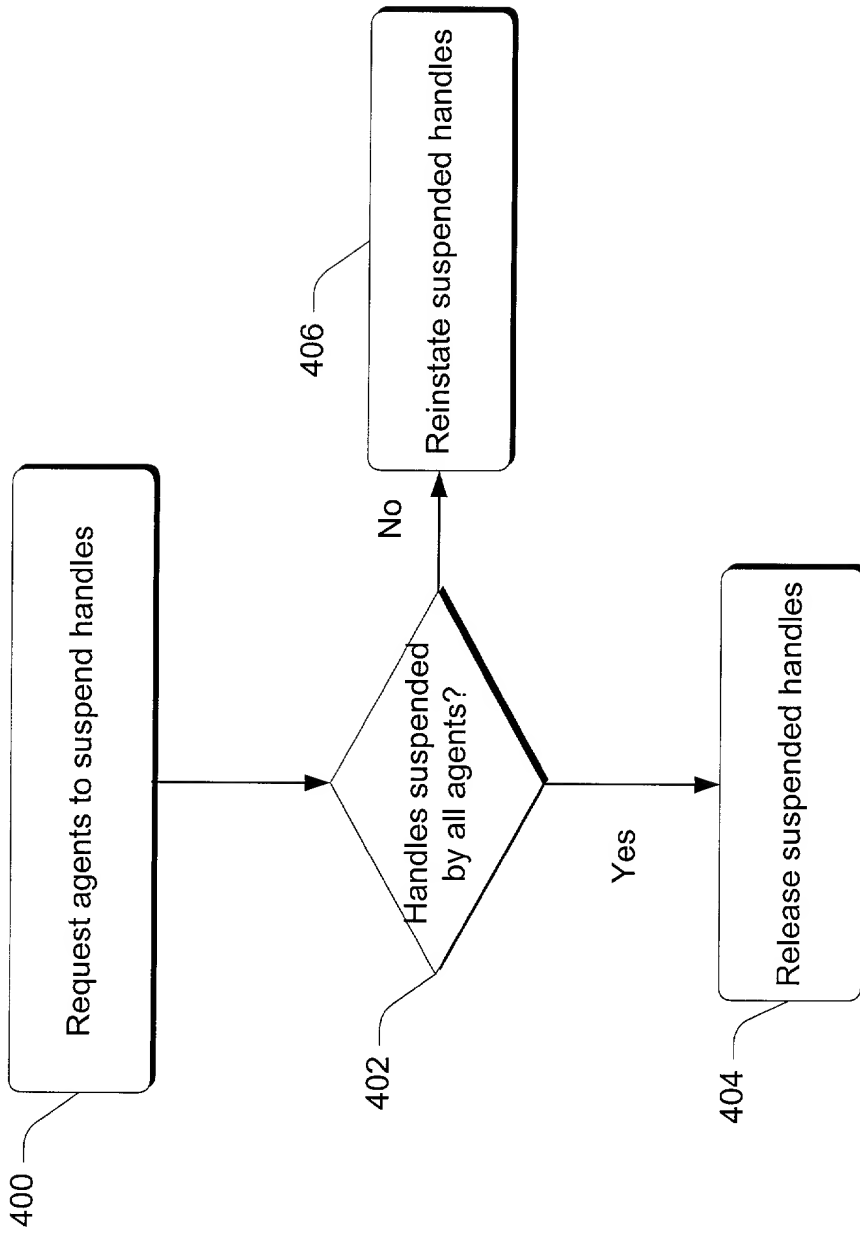


Fig. 4



500

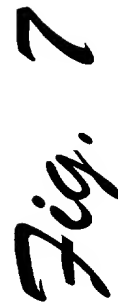
502	Type	Field	Description
504	Handle	handle	handle value
506	Resource*	resource	pointer to resource
	bool	suspended	Indicates suspended handle

Fig. 5

600

602	Index	handle	resource	suspended
	0	0	A	false
	1	1	B	false
	2	2	C	false
	3	3	D	true

Fig. 6



Index	handle	resource
0	0	A
1	1	B
2	2	C
3	$3 + 2^{10}$	D

Fig. 8



Fig. 9

Fig. 10